DNSSEC: HOW FAR HAVE WE COME?

Nick Sullivan
CloudFlare Inc., USA

Email nick@cloudflare.com

ABSTRACT

DNSSEC is a set of security extensions to DNS intended to provide a root of trust for DNS records. This paper is a summary of the state of the art in DNSSEC deployment and implementation on the Internet. We start with a description of Kaminsky's attack on DNS to motivate the need for trust in the DNS system. From here we describe some of the common arguments against DNSSEC, including NSEC and NSEC3 walking, and how DNSSEC can be an enabler for UDP reflection attacks. We then discuss useful extensions to DNSSEC, like DANE, and how these can be used to secure websites without trusting the certificate authority system. We also examine how far the effort has come in the decades since the technology was standardized, including adoption statistics and trends.

INTRODUCTION

The Domain Name System (DNS) is one of the oldest and most fundamental components of the modern Internet. As the mechanism that maps domain names to Internet Protocol (IP) addresses, it provides a human-readable layer to navigate the millions of machines and devices on the Internet. In the early 1980s, when DNS was designed, there was no strong need to add security mechanisms into the protocol. Computers of the time were underpowered compared with today's machines; public key cryptography was a relatively new concept; and the network was much smaller, with fewer participants who were relatively well known and trusted. As the network grew and evolved, DNS remained pretty much unchanged as an insecure and unauthenticated protocol.

In 1995, the IETF started a public discussion around how DNS could be made more trustworthy. Eventually, a set of extensions to DNS called Domain Name System Security Extensions (DNSSEC) were settled on and formally published in 2005, replacing earlier proposals as a definitive way forward for securing DNS. Though it has been almost a decade since this publication, DNSSEC still has a long way to go to be adopted in the mainstream.

There are several catalysts pushing DNSSEC adoption, one of which is Dan Kaminsky's cache poisoning attack from 2008 [1]. This attack highlighted the significant trust issues in traditional DNS, and how DNSSEC is well positioned to solve them. Even with the significant work going into it, there are several factors holding DNSSEC adoption back. Network operators tend to prefer stability to complexity (for good reason), and people have questioned whether DNSSEC is really the right tool for the job.

This paper presents the evolution of the state of the art in DNSSEC deployment and implementation on the Internet. We start with some background about how DNS works and where DNSSEC fits in. We will explore how standard DNS is insecure and can be exploited with Kaminsky's attack. We describe some of the common arguments against DNSSEC, including domain enumeration with NSEC (NextSECure) and NSEC3 walking, and how DNSSEC can be an enabler for UDP reflection attacks.

We then discuss useful extensions to DNSSEC, like DANE, and how these can be used to secure websites and provide a much stronger form of trust than the certificate authority system. We also examine how far the effort has come in the decade since the technology was standardized using adoption statistics and trends.

DNS: A DISTRIBUTED KEY VALUE STORE BEFORE IT WAS COOL

The concept of DNS is simple: it is a global database of information about domain names.

If a client wants to connect to an address such as 'www.example.com', and needs to know which IP address corresponds to this address, they can ask DNS. Typically, all DNS messages are sent over UDP.

There are several types of Resource Records (RR) that DNS can answer questions about. One of the most important RRs is called an A record, which stores the IPv4 address associated with the domain. To find out the value of a record for a given domain, you can get a tool called a DNS resolver to find out for you.

For example, it you wanted the IP for www.example.com, the question you could ask would be something like:

'What is the A record for the domain www.example.com?'

The raw DNS request is a UDP packet as shown in Listing 1. This request contains an ID (0x27e1), some flags to indicate that it is a request, and the question itself.

The response is shown in Listing 2. The response comes with the matching ID (0x27e1), a copy of the original request echoed back, some flags (1/0/0), and an answer to the question: 93.184.216.119. It also contains a validity period, called time to live (TTL), to indicate for how long the record is valid.

The ISP typically provides a nameserver that can answer DNS requests. You can use a public nameserver like the ones provided by *Google* (8.8.8.8, 8.8.4.4) or *OpenDNS* (208.67.222.222, 208.67.220.220). If a nameserver does not have the answer, it

```
0x0000: 27e1 0100 0001 0000 0000 0000 0765 7861 '............exa
0x0010: 6d70 6c65 0363 6f6d 0000 0100 01 mple.com.....
```

Listing 1: Raw DNS request.

```
0x0000: 27e1 8180 0001 0001 0000 0000 0765 7861 '............exa
0x0010: 6d70 6c65 0363 6f6d 0000 0100 01c0 0c00 mple.com........
0x0020: 0100 0100 0031 f500 045d b8d8 77 .....1...]..w
```

Listing 2: Response.

can request it from another server, called an authoritative nameserver, to get the proper resource record. Any nameserver that is willing to query other servers for DNS answers is called a recursive nameserver.

The hierarchy of DNS is very well defined: start at the DNS root nameserver and work your way down the domain name. The top of the pyramid is a server called the root server, the source of truth for all requests (there are several root servers maintained by different organizations [2]). The root servers know the addresses of the authoritative servers for the top level domains (.com, .edu, etc.), but nothing else. If you ask the .com authoritative nameserver for www.example.com, it will return a record called SOA (start of authority), which points to the nameserver which should have more information about the requested zone.

Each of the TLD authoritative nameservers knows about the authoritative nameservers for the domains under them (example.com, cloudflare.com, google.com, etc.). Following this downward eventually gets you to the server that can answer queries about the records for the subdomain you are looking for.

Root SOA → com authority → SOA example.com authority → A www.example.com

The set of DNS recursive and authoritative nameservers is one of the oldest distributed databases on the Internet. The TTL for RRs allow each server to act as a cache to help distribute the load. There are a number of different RRs that you can ask for. 'A' records result in IPv4 addresses, 'AAAA' for IPv6, 'SOA' for Start of Authority, and there are many more defined in various RFCs [3].

KAMINSKY'S ATTACK

In 2008 [4], Dan Kaminsky revealed an attack on the DNS system in which you can trick a DNS recursive nameserver into storing incorrect DNS records. Once the nameserver has stored the incorrect response, it will return it to everyone who asks as long as the TTL has not expired. This so-called DNS poisoning attack could allow arbitrary people to trick DNS and redirect web browsers to incorrect servers, allowing them to hijack traffic.

The attack is simple to describe but difficult to pull off. Take the sequence of events described in the last section:

1. Q client → recursive
2. Q recursive → authoritative
3. A authoritative → recursive
4. A recursive → client

The attack relies on the fact that UDP is a stateless protocol. Each of the requests and responses described here is a single UDP request containing to and from IP addresses. If the sender of the request lies about the return address, the response is sent to the wrong place. We will get back to this when we talk about reflection/amplification attacks. The main point here is that an attacker can create a DNS request or response with a forged from address.

Of the requests above, message number 3 looks like a good target to attack. The recursive nameserver will accept the first answer to its question. If you can answer it faster than the authoritative server can, it will accept your answer as truth. This is the core of the attack:

1. Pick a domain whose DNS you want to hijack.
2. Send a request to a recursive nameserver for the record you want to poison.
3. Send fake UDP responses pretending to be the authoritative server with the answer of your choosing (i.e. point the A record to an IP you control).

If your malicious response arrives ahead of the real response, the recursive nameserver will believe your record and cache it for as long as the TTL is set. Then any other clients asking for the poisoned record will be directed to your malicious domain.

There are some complications that make this harder than it sounds in practice. You have to guess the request ID, and you have to guess the incoming UDP port on the recursive nameserver. When Kaminsky's attack was originally proposed, these two values were easily guessable, making DNS poisoning a real threat. Since then, request ID and port randomization have made this specific attack more difficult to pull off, but not impossible.

The fundamental issue that allows this kind of attack to happen is that there is no way to validate that the records are what they are supposed to be.

DNSSEC

The security extensions to DNS add three important records that enable the entire DNS to be trusted. From a high level, DNS starts at the root nameserver and works down. This is the same way that trust is conferred via DNSSEC. The DNS root is the definitive root of trust, and a chain of trust is built to the root from any DNS entry. This is a lot like the chain of trust used to validate TLS/SSL certificates, except that, rather than many trusted root certificates, there is one trusted root key managed by the DNS root.

The point of DNSSEC is to provide a way for DNS records to be trusted by whoever receives them. The key innovation of DNSSEC is to use public key cryptography to ensure that DNS records are authentic. DNSSEC not only allows a DNS server to prove the authenticity of the records it returns, it also allows a server to assert the non-existence of records. This is established by a strict chain of records that identify either a public key or a signature of a set of resource records. The root of this chain of trust is the root key, maintained and managed by the operators of the DNS root.

There are several important new record types:

- DNSKEY: a public key, used to sign RR sets.
- DS: delegation signer, a hash of a key.
- RRSIG: a signature of a minimal set of resource records.

A DNSKEY record is a cryptographic public key, it usually comes in two flavours:

- KSK (key signing key): usually used to sign DNSKEY records only.

- ZSK (zone signing key): used to sign every other type of record.

So how do these records fit together to form a chain of trust? Let's examine first how an RRSIG record works. Each RRSIG record includes a cryptographic hash of all the records of a given type, sorted in a canonical order. It also includes a tag to help identify which DNSKEY record was used to sign it. For every record type other than DNSKEY, the RRSIG is signed with a ZSK DNSKEY. The DNSKEY is signed by the KSK. So by checking an RRSIG against the appropriate DNSKEY, you can assert that the appropriate server has signed the records and they are trusted.

How do you trust the DNSKEY? The way to do that is to walk the domain up to the next zone. To verify that the DNSKEY for example.com is valid, you have to ask the .com authoritative server. This is where the DS record comes into play. The .com zone stores the hash of the DNSKEY for each zone it knows about as a DS record. This record, like all DNSSEC records, is part of a signed set of records, and has an accompanying RRSIG signed by the .com DNSKEY. Similarly, the .com has a KSK, signing all .com DNSKEYs. So if you trust the .com keys, you can trust the example.com keys, and the A and other records obtained from the nameserver. The ultimate root of trust is the KSK DNSKEY for the DNS root. This key is universally known and published.

Here is the DNSKEY root KSK as of June 2014:

```
AwEAAagAIKlVZrpC6Ia7gEzahOR+9W29euxhJhVVLOyQbSEW0O8gcC
jFFVQUTf6v58fLjwBd0YI0EzrAcQqBGCzh/RStIoO8g0NfnfL2MTJR
kxoXbfDaUeVPQuYEhg37NZWAJQ9VnMVDxP/VHL496M/QZxkjf5/Efu
cp2gaDX6RS6CXpoY68LsvPVjR0ZSwzz1apAzvN9dlzEheX7ICJBBtuA
6G3LQpzW5hOA2hzCTMjJPJ8LbqF6dsV6DoBQzgul0sGIcGOYl7OyQd
XfZ57relSQageu+ipAdTTJ25AsRTAoub8ONGcLmqrAmRLKBP1dfwhY
B4N7knNnulqQxA+Uk1ihz0=
```

By following the chain of DNSKEY, DS and RRSIG records to the root, any record can be trusted.

These records are enough to prove the existence of a given resource record, but something more is needed in order to prove that a record does not exist. This is where two additional record types, NSEC and NSEC3, come into play.

Typically, a DNS authoritative server returns a flag called NXDOMAIN when a resource record definitively does not exist. In practice, this is not enough to prove that the record does not exist, since the response could be forged by a third party with no additional knowledge. NSEC works by signing the gaps between valid responses. There are a finite set of positive answers, and the gaps between positive answers are covered by NSEC. This effectively doubles the number of records in the zone, but allows an authoritative nameserver to serve a signed response for any question.

Say that .com supports NSEC records (in reality it supports NSEC3). Asking for 'example.com' would give you a positive answer with an IP address and an RRSIG record. Asking for 'examplf.com' would give you a negative answer 'there are no zones between example.com and exampli.com', with a corresponding RRSIG.

For NSEC3 the same logic applies, but for the hash of the record. Asking for examplf.com (e comes before f) would give you 'there are no records with hashes between A and B', where A is the next closest hash lexicographically before the hash of examplf.com, and B is the next closest after.

DNSSEC CONTROVERSIES

As a way to provide a system of trust for DNS records, DNSSEC works. Implementing such a large change to one of the core components of the Internet was not expected to be easy, and it has not been. Not only is there a large operational cost to implementing DNSSEC, there have been several other criticisms faced by DNSSEC over the years. In this section I will examine two major drawbacks to DNSSSEC.

Zone privacy

One of the major issues with the NSEC record is that it allows anyone to 'walk' the DNS zone. This can be used to reveal all of the domains hosted on the nameserver. Take the example of 'examplf.com' – the NSEC responses lets you know that 'exampli.com' exists. Trying an invalid record after that one such as 'examplia.com' will give you the next valid DNS name. This can continue until the entire zone is enumerated.

Technically, DNS records are not supposed to be secret, but in practice they are considered so. Subdomains have been used to keep things private for a while, and suddenly revealing the contents of the zone file is not expected or appreciated. This is why NSEC3 was introduced, but even it can be used to reveal the existence of subdomains.

The NSEC3 record is designed to provide authenticated denial of existence by having a signed ring of password equivalents. Rather than a signed gap of domain names for which there are no answers to the question, NSEC3 provides a signed gap of hashes of domain names. This was intended to prevent zone enumeration.

For NSEC enumeration, you can create the full list of domains by starting at a given known domain. If the zone has around 100 domains, it will take around 100 requests to enumerate the entire zone. With NSEC3, when you request a zone that does not exist, a signed NSEC3 record is returned with the next zone present ordered lexicographically by hash. By pre-computing domains that hash to the next gap of domain hashes, it should take around 100 correctly chosen queries to enumerate all the hashes. There are many tools that can do this computation for you, including a plug-in to nmap [5]. Once the hashes that correspond to all the valid subdomains of the zone are known, a dictionary attack can commence. Subdomains that have a short enough name, or are guessable using a dictionary, can be revealed as existing without having to spam the nameserver with guesses. Zone privacy is only slightly improved when using NSEC3 as designed.

This vulnerability is mitigated by a technique introduced by Dan Kaminsky known as DNSSEC White Lies [6]. When a request comes in for a domain that is not present, instead of providing an NSEC3 record of the next real domain, an NSEC3 record of the next hash lexicographically is presented. This does not break the NSEC3 guarantee that there are no domains whose hash fits lexicographically between the NSEC3 response and the question.

The question of implementing NSEC3 White Lies or not comes down to whether a signature can be computed on the fly. The traditional creation of zone records to be served with DNS resolution happens offline: a set of records is saved into a file format like BIND and used by the live DNS server to answer questions. Having a DNS server with the minimum amount of logic inside allows the operator to conserve CPU resources. In order to do NSEC3 White Lies, the records need to be generated dynamically based on the request. This changes the operational conditions needed to support DNSSEC: the DNS authoritative server itself needs to do the cryptographic operations in response to the incoming query. This demand for live signing imposes several other security problems in distributed environments.

Key management

As described above, NSEC and NSEC3 were designed for the offline world. A DNS zone is often a static file with all signatures pre-computed on a machine that is not connected to the network. If you remove that requirement and allow cryptographic operations to happen live, you get some valuable tools like NSEC3 White Lies. The downside is that your key material is now live on a machine connected to a network. The main problems with live signing are key security and resource utilization.

Recently, the world was shocked by a bug known as Heartbleed that opened up a major security hole in server applications. It was caused by a coding error in OpenSSL that exposed a remote memory disclosure vulnerability. By sending a carefully constructed packet to a server that uses a version of OpenSSL with the flaw, an attacker could reveal server memory. In particular, it is possible to extract the TLS private key from the server without a lot of effort. Heartbleed-style bugs are just one of the many threats to private key security when the key is being used in an active process. The more a machine is exposed to the Internet, the more vectors of attack there are. Offline signing machines have a much smaller window of exposure to threats.

One way to keep keys secure is to use a hardware-backed solution such as a hardware security module (HSM). One of the major drawbacks for this is cost – HSMs are very expensive. This is one of the stickiest points for running DNS servers that are spread out geographically in order to be close to their customers. Running an HSM in every server location can not only be expensive, but there can be legal complications as well [7].

Reflection/amplification threat

Another fear for operators running an authoritative DNS server is that they will be used as a vehicle for malicious distributed denial of server (DDoS) attacks. This stems from the fact that DNS uses UDP, a stateless protocol.

In TCP, each connection begins with a three-way handshake. This ensures that the IP address of both parties is known and correct before starting a connection. In UDP, there is no such handshake, messages are just sent directly to an IP with an unverified 'from' IP address. If an attacker can craft a UDP packet that says 'hi, from IP X' to a server, the server will typically respond by sending a UDP packet to X. Choosing X as a victim's IP address instead of the sender's is called UDP 'spoofing'. By spoofing a victim, an attacker can cause a server that responds to UDP requests to flood the victim with 'reflected' traffic.

DNSSEC also works over UDP, and the answers to DNS queries can be very long, containing multiple DNSKEY and RRSIG records. This is a big target for attackers since it allows them to 'amplify' their reflection attacks. If a small volume of spoofed UDP DNSSEC requests is sent to a nameserver, the victim will receive a large volume of reflected traffic. Sometimes this is enough to overwhelm the victim's server, and cause a denial of service.

Like many services, DNS can also work over TCP. There is a flag called the 'truncation' flag that can be sent back to a resolver to indicate that TCP is required. This would fix the issue of DNS reflection at the cost of slower DNS requests. This solution is not practical at the moment since 16% of resolvers don't respect the TCP truncation flag, and 4% don't try a second server [8]. TCP is not the solution at the moment. Using resource rate limiting and using proper heuristics into servers seems to be the best way to mitigate this threat at the moment.

VALUABLE EXTENSIONS

The security provided by DNSSEC is helpful for securing regular DNS records against poisoning attacks or other malicious manipulation. Having a secure root of trust is also useful for securing other systems. One interesting application of this is providing an alternative root of trust for the standard web PKI.

Websites are typically secured through a mechanism called Transport Layer Security (TLS). When a browser accesses a site over the HTTPS protocol, the web server authenticates itself to the client using its TLS certificate and the two parties establish a shared key for further encrypted communication. In order to prove authenticity of the server's certificate, it is digitally signed by a third party known as a Certificate Authority (CA). To ensure that the site is who it says it is, the browser validates the site's certificate against the domain name of the site and ensures that the certificate authority that was used to sign the certificate is trusted.

Different browsers on different platforms have different lists of CAs that they trust. For example, *Windows 8.1* trusts 227 so-called root certificates, and *Firefox* (version 30) only trusts 143 certificates. One of the main criticisms of web PKI is that there are too many trusted certificate authorities. In 2011, a CA called *TURKTRUST* mis-issued an intermediate certificate [9] that allowed rogue entities to create certificates that were valid for google.com and other high-profile web domains [10]. In the same year, another CA called *Diginotar* was found to have been compromised by attackers [11]. Clearly trusting so many different entities is dangerous for web browsing.

Enter DANE (DNS-based Authentication of Named Entities), an extension of TLS that allows a website owner to embed its TLS certificate into the DNS. With DNSSEC signatures used to validate the correctness of the certificate, it is no longer necessary for the browser to trust the CA, they can trust DNS

instead. With one heavily protected root of trust, website identities are much more difficult to subvert.

Defined in RFC 6698 [12], DANE adds a new resource record to DNS: TLSA. This new record can be used to indicate the expected certificate for a given domain, or identify which CA should be expected as the root of trust for the certificate. If DANE had been implemented and validated by browsers during the *TURKTRUST* incident, attackers would not have been able to spoof google.com because they not only had a new certificate, but the one presented was signed by the wrong certificate authority (google.com's certificate is signed by *GeoTrust*).

DEPLOYMENT STATISTICS

It has been more than nine years since the publication of the DNSSEC RFCs and, needless to say, deployment is far from complete. There are many different components involved. We will summarize the progress in each of the following areas:

- Signing the root zone
- Signing the TLD zones
- Registrar integration
- Validating resolvers

Signing the root

The first step in getting DNSSEC to work is signing the root. This is the most important piece of DNSSEC as this key forms the root of trust for the entire system.

ICANN has specified a set of instructions for managing the root keys and signing the root zone [13]. This is called a key ceremony and involves many of the original creators of the Internet. The first root zone keys were published on 15 July 2010, just after the first and second key-signing ceremonies.

Signing the TLDs

After the signature of the DNS root zone, the next step is signing and managing the keys for the various top level domains (.com, .net, .gov, etc.). The first generic TLD (gTLD) to be signed was .org in June 2009 [14]. As of 29 June 2014, 445 out of 630 TLDs in the root zone are signed [15]. The full list has been visualized by .se's *DNSCHECK* tool [16].

Figure 1 shows the number of TLDs that have registered their keys with the root [17]. There was a large increase starting in late 2013 when many more gTLDs were added, all of which were required to implement DNSSEC from the outset [18].

Recently, the TLD for the European Union, .eu, presented the history of its deployment of DNSSEC [19]. The .eu registry started accepting DNSKEY material in June 2010, and got the .eu KSK's DS record signed by the root in September 2010. By May 2014, 267,000 of 3,840,000 (6.9%) .eu zones had been signed. The progress is shown in Figure 2.

Another large TLD with DNSSEC enabled is .cn [20], which updated its zone's DS in the root in September 2013 and became fully operational in December of that year. For this zone, the keys are kept by five key administrators who are each

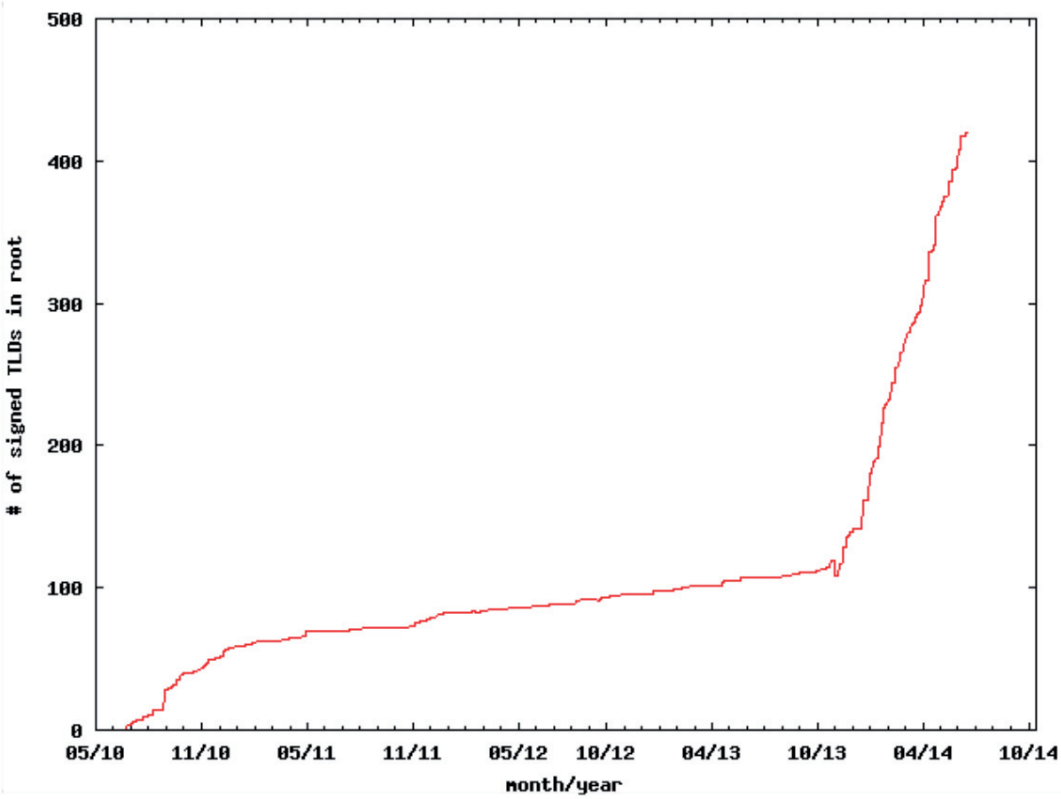

Figure 1: Number of TLDs that have registered their keys with the root.

Figure 2: Progress of .eu domain signing.

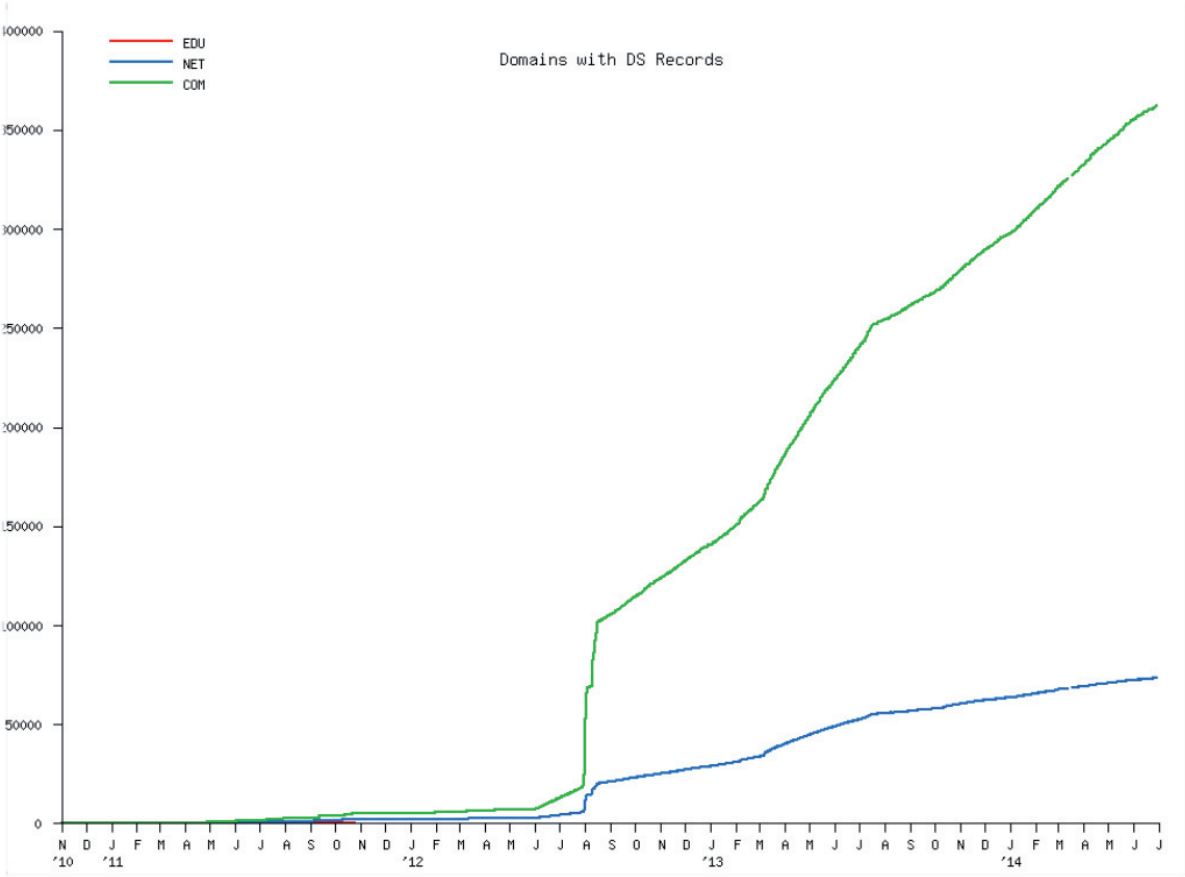

Figure 3: DNSSEC adopion continues at a steady rate – surpassing 350,000 .com domains in 2014.

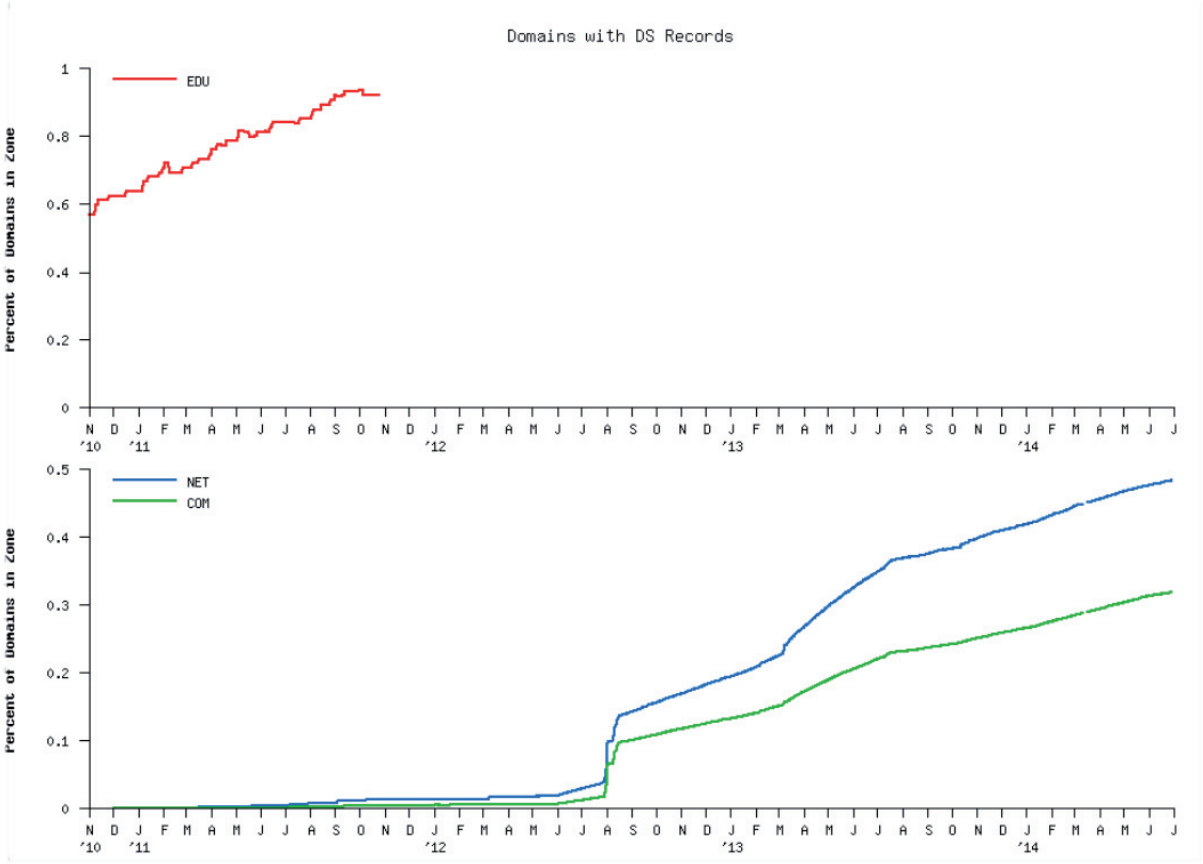

Figure 4: Less than 0.5% of domains in either the .com or .net zones are accounted for.

given a smart card containing a segment of the keys. The full key can be constructed from any three of the five segments.

The process is still young, and each TLD is developing its own set of standards and practices. ICANN has published RFC 4641 [21] to help explain and standardize the process, including expiration and validity periods for keys. For the .cn keys, the following approach was taken:

ZSK: RSA-SHA256 1024-bit, NSEC3

KSK: RSA-SHA256 2048-bit, NSEC3

Key rotation cycle:

ZSK: 100 days, RRSIG period: 30 days

KSK: 13 months, RRSIG period: 30 days

Going back to 2012, DNSSEC deployment was much less advanced [22], only 86/313 TLDs were signed. As shown in Figure 3, adoption is continuing at steady rate, surpassing 350,000 .com domains in 2014 [23]. It should be noted that this accounts for less than 0.5% of domains in either the .com or .net zones, as Figure 4 shows [24].

Registrar integration

Part of the DNSSEC process for domains is getting their DS record into the proper TLD zone and signed. The KSK can be created and added to a zone, but without a DS record there is no way for resolvers to validate or connect the signatures to the root. This is handled by the site's registrar.

As of 27 May 2014, 35 registrars can accept DS records. This includes many leading registrars such as *GoDaddy*, *DYN* and *OVH*. Several of these registrars also handle signing services if needed. A full list is provided by ICANN [25]. As of 2013, all new registrar applicants are required to support DNSSEC [26].

Validating resolvers

Even with all of the root, TLD, and authoritative servers serving valid DNSSEC records, they need to be validated by resolvers to have any weight.

Comcast was one of the first major services to validate DNSSEC records by default. All of *Comcast*'s domains were signed and customer DNSSEC validation was turned on in January 2012 [27]. *Google Public DNS* is another popular DNS resolver service that started validating DNSSEC responses by default on 6 May 2013 [28].

CONCLUSIONS

DNSSEC is a valuable tool for improving the trust and integrity of DNS, the backbone of the modern Internet. DNSSEC

deployment is still in its infancy, less than five per cent of all zones had been signed as of mid-2014. Though is has its detractors, adoption is increasing and DNSSEC is becoming a core tool in the development of a safer and more trustworthy Internet.

REFERENCES

[1] https://kb.isc.org/article/AA-00924/0/CVE-2008-1447%3A-DNS-Cache-Poisoning-Issue-Kaminsky-bug.html.

[2] http://www.iana.org/domains/root/servers.

[3] http://www.zoneedit.com/doc/rfc/.

[4] http://spectrum.ieee.org/images/oct08/images/phish03.pdf.

[5] http://nmap.org/nsedoc/scripts/dns-nsec3-enum.html.

[6] http://dankaminsky.com/2011/01/05/djb-ccc/#whitelies.

[7] http://www.kslaw.com/imageserver/KSPublic/library/publication/2011articles/ 11-11WorldECRCloutierCohen.pdf.

[8] http://www.circleid.com/posts/20130820_a_question_of_dns_protocols/.

[9] http://nakedsecurity.sophos.com/2013/01/08/the-turktrust-ssl-certificate-fiasco-what-happenedand-what-happens-next/.

[10] https://krebsonsecurity.com/2013/01/turkish-registrar-enabled-phishers-to-spoof-google/.

[11] http://threatpost.com/final-report-diginotar-hack-shows-total-compromise-caservers-103112/77170.

[12] http://tools.ietf.org/html/rfc6698.

[13] http://www.root-dnssec.org/wp-content/uploads/2010/02/draft-icann-dnssec-ceremonies-00.txt.

[14] http://www.circleid.com/posts/20090602_org_first_open_top_level_domain_dnssec/.

[15] http://stats.research.icann.org/dns/tld_report/.

[16] http://tldwithdnssecandipv6.se/.

[17] http://rick.eng.br/dnssecstat/.

[18] https://www.dnssec-deployment.org/index.php/2012/01/new-gtlds-will-support-dnssec-fromthe-start/.

[19] https://london50.icann.org/en/schedule/wed-dnssec/presentation-dnssec-eu-25jun14-en.

[20] http://www.apirc.org/previous/2013/wjxz/201309/P020130930491213318448.pdf.

[21] http://www.ietf.org/rfc/rfc4641.txt.

[22] https://www.icann.org/en/system/files/files/menog-dnssec-deployment-30apr12-en.pdf.

[23] http://scoreboard.verisignlabs.com/count-trace.png.

[24] http://scoreboard.verisignlabs.com/percent-trace.png.

[25] https://www.icann.org/resources/pages/deployment-2012-02-25-en.

[26] http://www.internetsociety.org/deploy360/blog/2013/09/icanns-2013-raa-requires-domainname-registrars-to-support-dnssec-ipv6/.

[27] http://www.internetsociety.org/deploy360/resources/case-study-comcasts-dnssecimplementation/.

[28] https://groups.google.com/forum/?hl=en#!topic/public-dns-announce/67oxFjSLeUM.